**BMP Image Edge Detector**

**Phase 2 Proposal**

Submission Date: 2/24/2018

Thursday 14:30 Lab (Lab Section 11)

TA: Reena Elangovan

**Prepared by:**

Ni Kang

Haoming Zhang

Haobo Chen

Graham Sninsky

**1. Executive Summary**

Edge detection is a basic problem for image processing and computer visualization, whose aim is to denote the points which change obviously in digital image. The obvious changes in image properties usually reflect the property’s important events and changes, which includes: discontinuity in depth, discontinuity on surface, the changes in matter properties and the brightness change in a specific scenario. Image edge detection drastically decreases the number of data and eliminates the irrelevant information which keeps the important properties of the image structure. There are a lot of ways to be used in edge detection, and they can be divided into two parts. The first part is to find the maximum and minimum value in the first derivation of the image. The second part is to find the edge through find the second derivation of the image. This result in a large amount of arithmetic and repeated processing, which, makes an ASIC design a perfect fit for the purpose. In addition, having a SoC design produced specifically for them function described above will drastically diminish the workload for processing images.

The design will utilize AHB Lite Interface, which is capable for both reading and writing. The AHB Lite is required for communication between systems in the design. Since edge detection requires large scale computation, AHB Lite itself won’t be able to store all necessary data. All data will be stored in registers in order for edge detection. The edge detector will also have two dedicated core, one for image filtering, the other is for edge detection.

The successful design of the proposed edge detector will require the following resources:

* AHB Lite Datasheet
* Reference Standard Cell Simulation Library for Mapped Design Verification
* Reference Standard Cell Technology Library for Final Design Layout Verification
* Verilog HDL Simulation and Design Synthesis Tool Chain

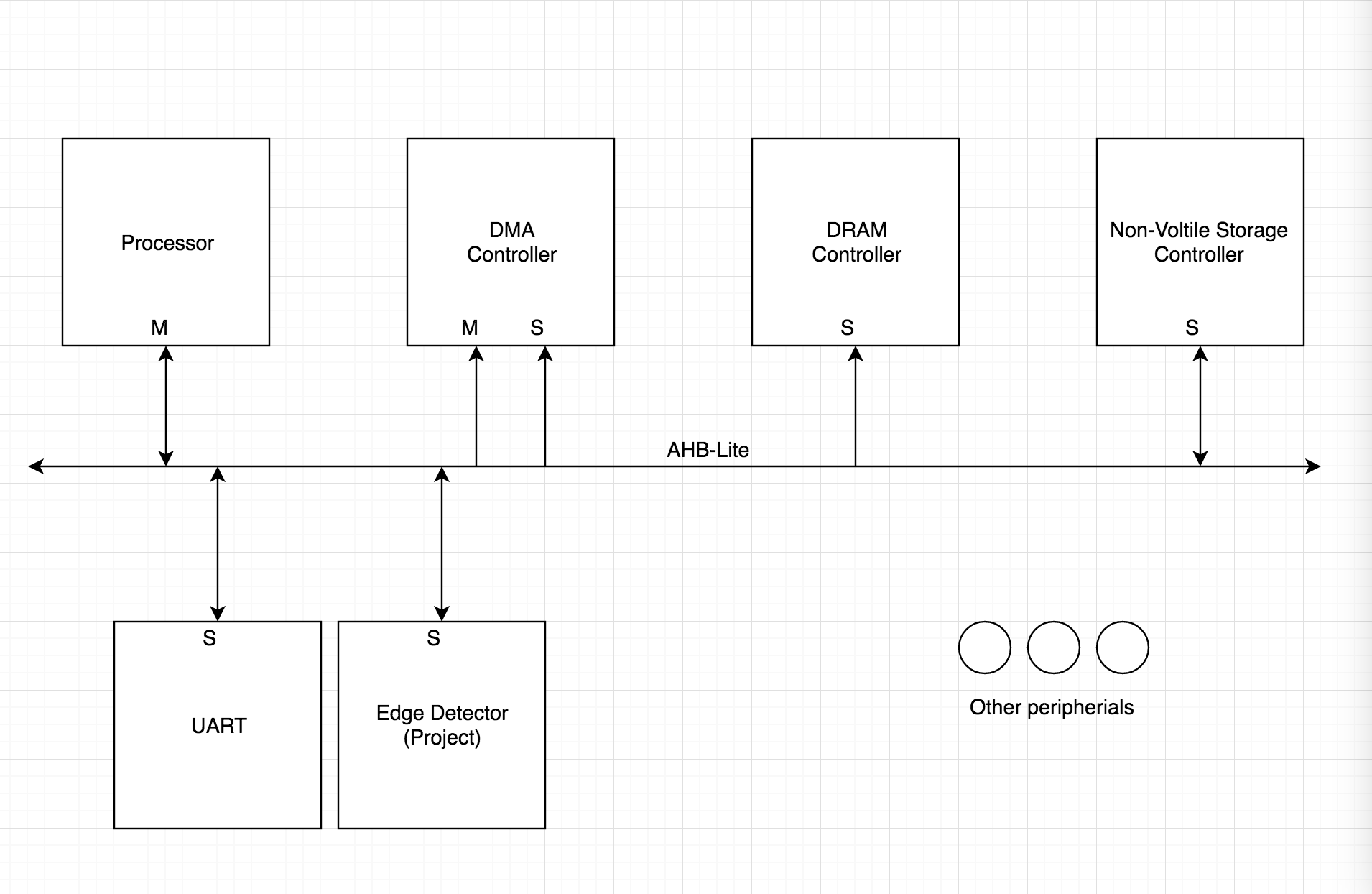
The following documents will describe:

* Super high level block diagram of edge detection
* Edge detector address mapping
* Design pinout
* Design architecture

**2. Design Specifications**

***2.1. System Usage***

***2.1.1. System Usage Diagram***

******

*Figure 1: Example System Usage Diagram for Edge Detector*

An example system illustrating the intended use of this edge detector is depicted above in Figure 1. In this

system there is a main processor where any relevant software would be executed, volatile main memory,

non-volatile storage, an UART module designed to serially input the image, our edge detector, and Direct Memory Access (DMA) controller for handling large data movement between slaves. The key operational ideas are that the software running on the processor would perform the following steps:

1. Configure the operational settings directly on the edge detector

2. Schedule a DMA transfer to populate the filter values

3. Directly send the image data to the edge detector

4. Wait until edge detector’s completion status bit is asserted

5. Read the calculation result for use

***2.1.2. Implemented Standard(s) and Algorithm(s)***

**Edge Detector**

18-bit Integer Filter Values

Configurable for signed or unsigned value usage

Default configuration is signed

36 \* 3-byte Integer Result Magnitude values

Configurable for signed or unsigned value usage

Default configuration is signed

**AHB Lite Standard Slave**

48-bit data bus

Read and Write Transfers

Burst Transfers Supported

Pipelined Transfers

10-bit, 522 word-address namespace (0x000-> 0x304)

***2.1.3. Edge Detector Namespace Address Mapping***

*Table 1: Namespace Address Mapping Table*

|  |  |  |  |
| --- | --- | --- | --- |
| Slave Word | Address Read / Write | Data Size (Byte) | Description |
| 0x000 - 0x0FF | R/W | 2 | Input filter matrix data |
| 0x100 - 0x1FF | R | 4 | 64 bit accumulator |
| 0x200 | R/W | 1 | Configuration Register 0: Number of samples for calculation |
| 0x201 | R | 1 | Status Register:  Bit 0: filter done (1 -> true, 0 -> false)  Bit 1: result done (1 -> true, 0 -> false) |
| 0x202 | R/W | 1 | Control Register:  Bit 0: clear prior sample (1 -> clear, 0 -> keep)  Bit 1: clear prior filter (1 -> clear, 0 -> keep) |

***2.2. Design Pinout***

*Table 2: Miscellaneous Pinout Table*

|  |  |  |  |
| --- | --- | --- | --- |
| Signal Name | Direction | Number of bits | Description |
| vcc | PWR | / | Power Pin |
| gnd | GND | / | Ground Pin |
| clk | IN | 1 | System clock(100MHz) |
| n\_rst | IN | 1 | Asynchronous Reset. (Active Low) |

*Table 3: AHB Lite Interface Pins*

|  |  |  |  |
| --- | --- | --- | --- |
| Signal Name | Direction | Number of bits | Description |
| header\_detected | IN | 1 | Input image header read. If the input image is less than 3x3 or larger than 6x6, set it to 0. Else, set it to 1. |
| image\_data | IN | 24 | Single pixel image input |
| filter | IN | 18 | Sobel filter value |
| process\_done | OUT | 1 | Set to high when process finished |
| image\_out | OUT | 24 | Single pixel image output |
| data\_ready | IN | 1 | Set to high when data is ready to be processed |
| one\_k\_samples | OUT | 1 | Active high signal indicating that 1,000 samples have been processed since the last assertion/power on. |

***2.3. Operational Characteristics***

**2.3.1. Gradient Calculating Operation**

The edge detection is applying the kernels similar to the following filter kernels on each pixel of the image.

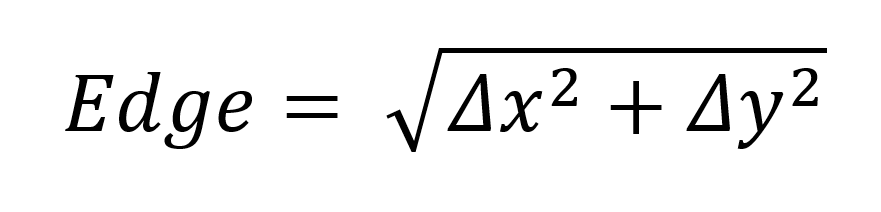
|  |  |  |
| --- | --- | --- |
| **-1** | **0** | **1** |

|  |
| --- |
| **-1** |
| **0** |
| **1** |

By applying the filter kernel in both horizontally and vertically, we will be able to find the gradient relation between the pixel of interest and its surrounding pixels in both directions.

**2.3.2. Edge Detecting Operation**

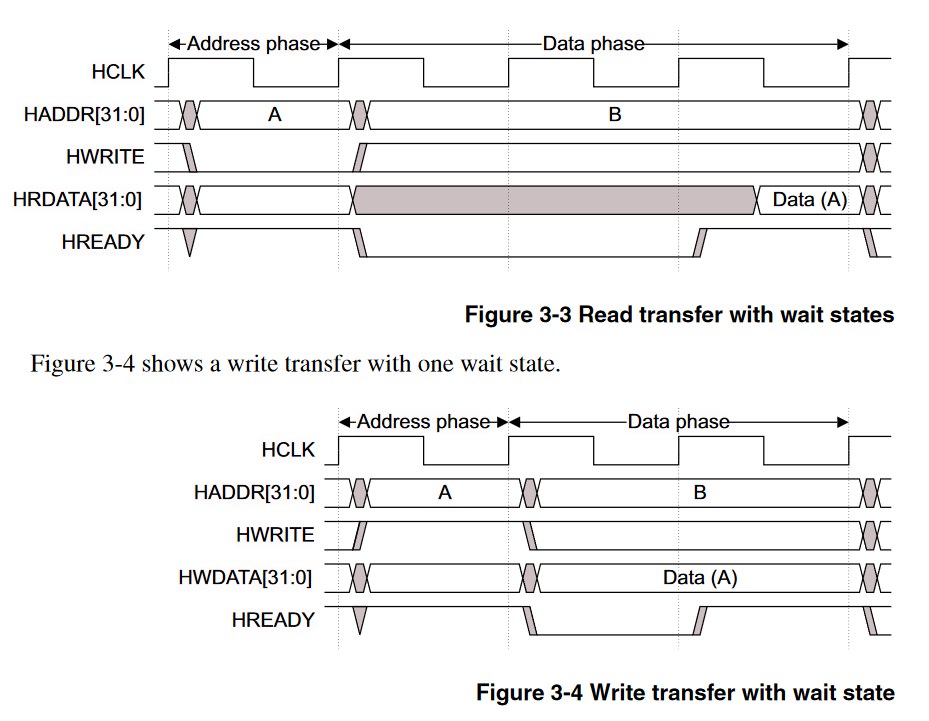
After a group of gradient is found, the Pythagorean theorem is applied to the two gradient vectors. Since it is possible to have a negative gradient, it is necessary to take the absolute values of them.



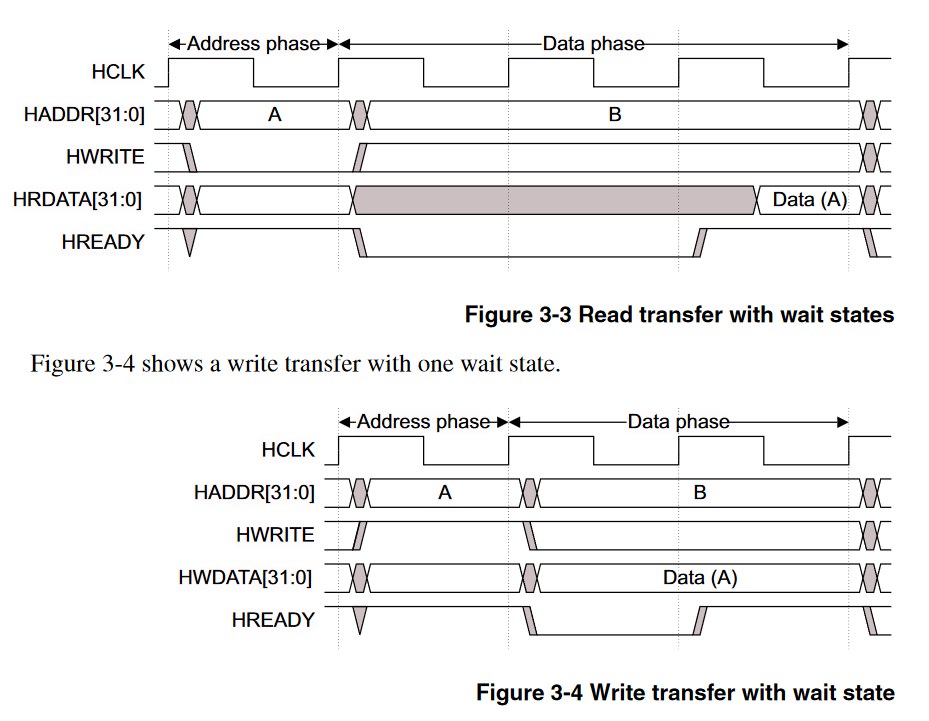
If the edge value is larger than a threshold of our choice, it is recognized as a valid edge and will output FF (ie. white pixel). Otherwise it will output 00 (ie. black pixel).

By performing the operation above, an image with only major edges will be generated.

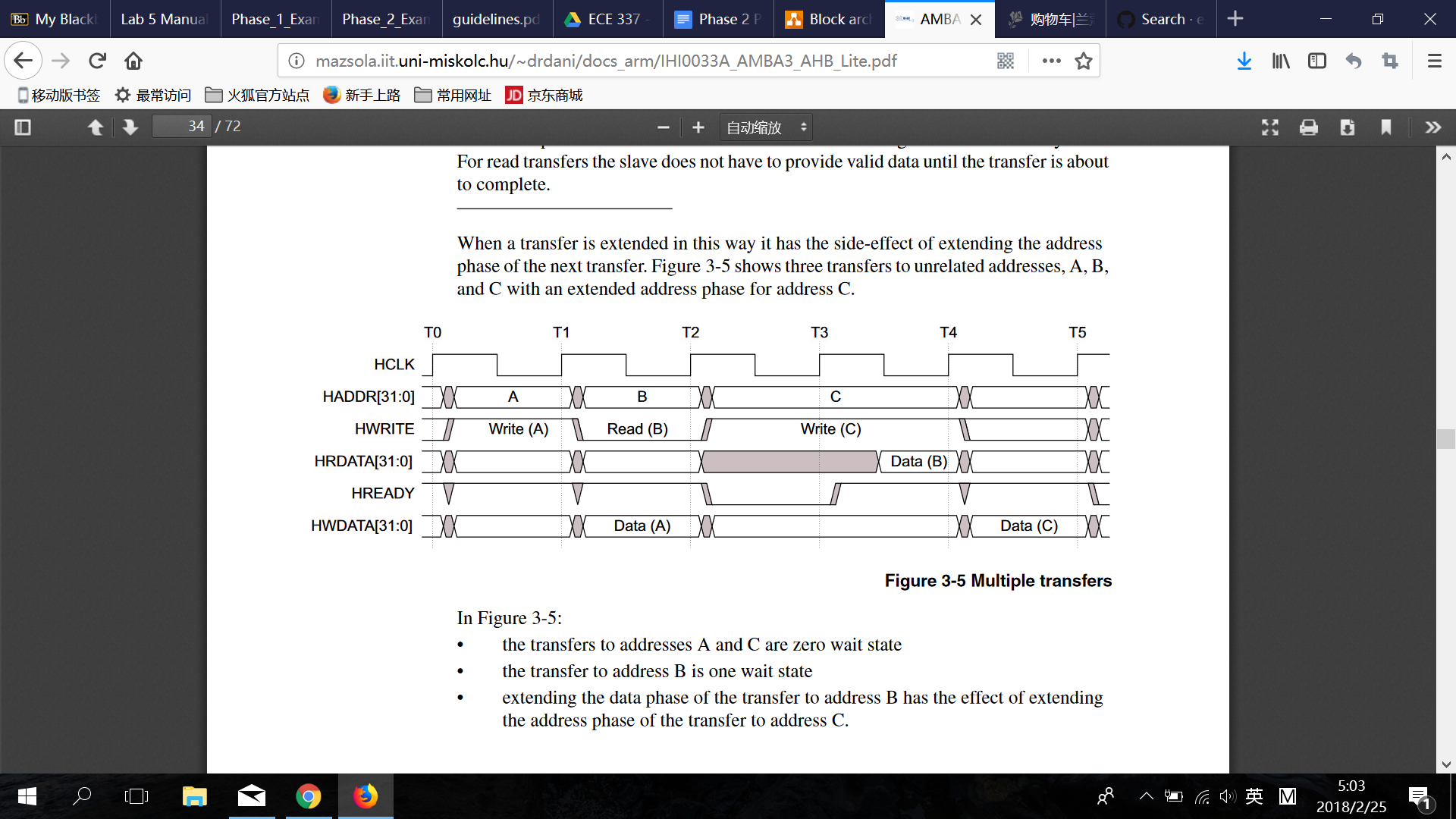
***2.3.3. Supported AHB-Lite SoC Bus Transactions***

***2.3.3.1. Single Word Transfers with wait***

*Figure 2: AHB Lite module single word read with wait timing diagram (adapted from AHB Lite Interface Specifications manual)*

****

*Figure 3: AHB Lite module single word write with wait timing diagram (adapted from AHB Lite Interface Specifications manual)*



*Figure 4: AHB Lite module multiple transfers timing diagram (adapted from AHB Lite Interface Specifications manual)*

Read Transfer

The HWRITE signal indicates both read and write state. During single word read transfers, when the HWRITE set to low, the system begins to read data from HADDR.

Write Transfer

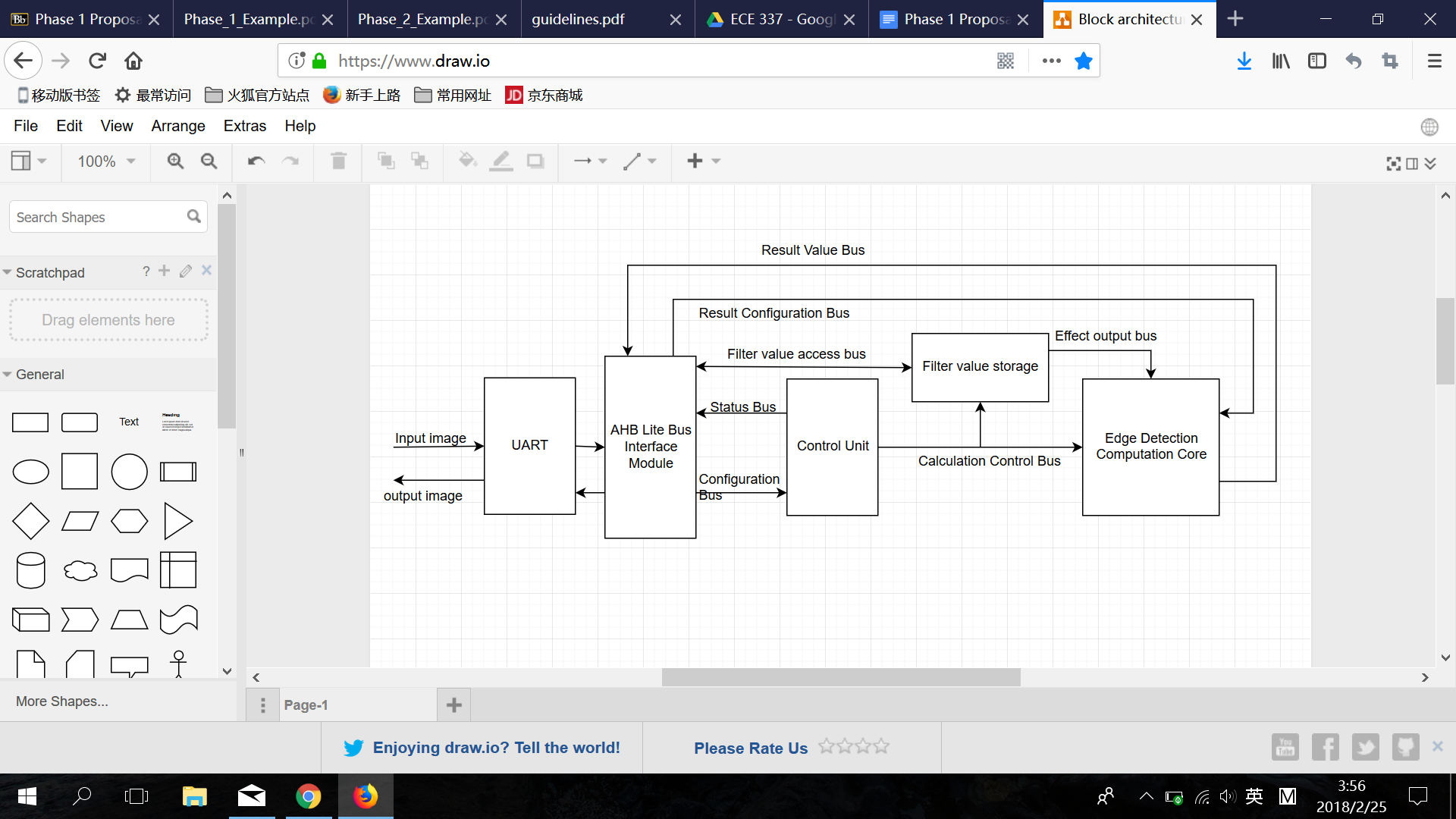
After finishing the read process, the HWRITE flag set to high into write state and write the new data into HWDATA. In the meanwhile, wait for the read data from the previous read process and read into HRDATA.

***2.4. Requirements for Design***

When implementing the edge detector, the choice of threshold number is critical. The actual choice will be tested during the experiment. Applying the different thresholds of the design will affect the final output rate. If a large threshold is chosen, the output will be vague (the edges of the picture will not be shown clearly, there might be missing parts if the color gradient is not large enough). Otherwise, if the threshold is low, the output will be a mess (the edge will include redundant edges, which will not perfectly outline the image). Thus, we have to find the the perfect threshold of the bmp edge detector through different tests, which is also the first requirement of our design. Secondly, knowing the size of the output image is also critical. We have a one thousand samples flag to measure how many bytes were passed into and processed by the system. This flag will not only be used in the purpose of indicating how many bytes are passed in, but will also be implemented for debugging purpose. This is the second main requirement for our design.

**3. Design Implementation**

***3.1. Design Architecture***

******

*Figure 5: Edge Detection Architecture Diagram*

The intended implementation architecture is depicted above in Figure 2. A module handling the

implementation of the AHB Lite interface is used to bridge between the

external SoC environment and the internal data and status locations. There will be a UART module which is used to serially input the input image data. There will be one value storage module, which is for the filter value set. It will be implemented using the underlying stream register design that will be defined in subsequent documents. A edge detection computation core module will house all of the arithmetic related portion of the design for processing a stage of the calculation. These two components (filter storage, and computation core) will be stepped through the full algorithmic edge detection calculation for new image data by the main control unit, which will be implemented as a Finite-State-Machine (FSM) with subordinate counters for tracking the progression through the needed stages.

**Reference Cited:**

“AMBA® 3 AHB-Lite Protocol,” 2006. [Online]. Available: http://mazsola.iit.uni-miskolc.hu/~drdani/docs\_arm/IHI0033A\_AMBA3\_AHB\_Lite.pdf. [Accessed: 24-Feb-2018].